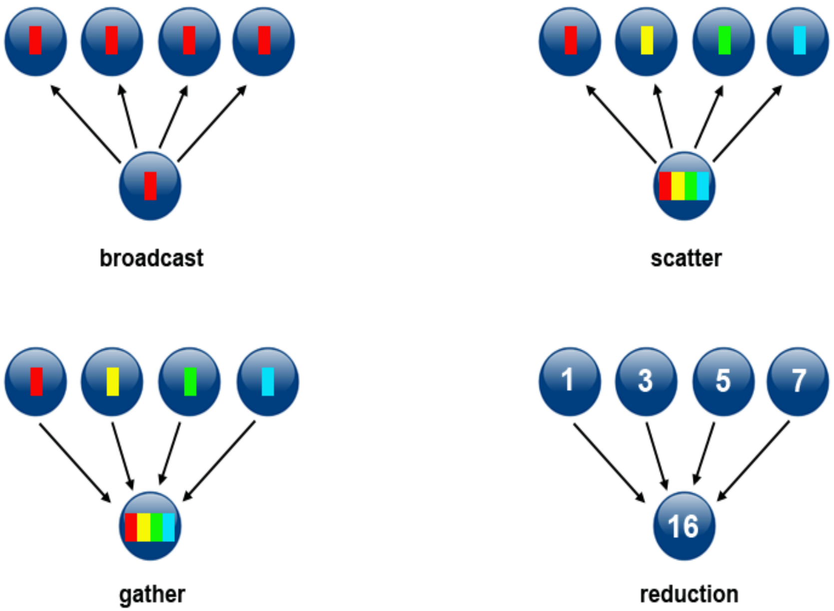
Exercise sheet – MPI

Simon Scheidegger
simon.scheidegger@gmail.com
Sept 20th, 2018
Cowles Foundation – Yale

Supplementary material for the exercises is provided in YaleParallel2018/day3/code/MPI/supplementary_material_mpi

Including adapted teaching material from of G. Hager & G. Wellein, B. Barney and the Swiss Supercomputing Centre (CSCS)

Reminder – collective communication



1. MPI exercise – Bcast

- Have a look at the code:
  YaleParallel2018/day3/code/MPI/supplementary_material_mpi/broadcast.f90 or
  broadcast.cpp

- write a makefile to compile the code.

- hard code a value (any double precision value), and broadcast the value of rank 0 to all Ranks.

- submit the job via slurm.

2. MPI exercise – Allreduce

- Have a look at the code:

YaleParallel2018/day3/code/MPI/supplementary_material_mpi/allreduce.f90
 or allreduce.cpp

- write a makefile to compile the code.

- calculate the sum of all ranks.

- submit the job via slurm.

3. MPI exercise – Scatter

- Have a look at the code:
  YaleParallel2018/day3/code/MPI/supplementary_material_mpi/scatter.f90
  or scatter.cpp

- write a makefile to compile the code.

- scatter the value of *senddata* of rank 0 to *receivedata* of all ranks.

- submit the job via slurm.

4. MPI Exercise (I)

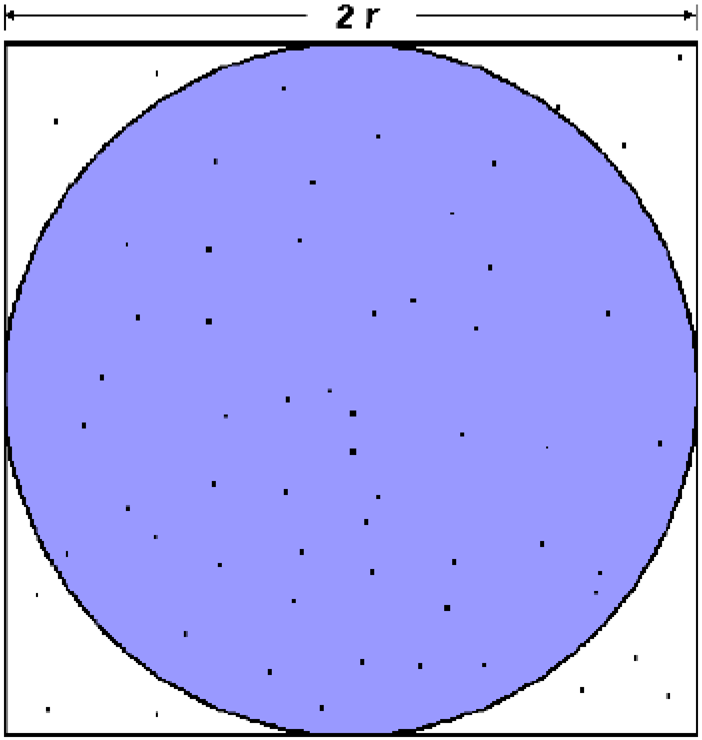
Revisit the following method of approximating PI:

1. Inscribe a circle in a square.

2. Randomly generate points in the square.

3. Determine the number of points in the square that are also in the circle.

4. approximate PI.



$A_S = (2r)^2 = 4r^2$

$A_C = \pi r^2$

$\pi = 4 \times \frac{A_C}{A_S}$

4. MPI exercise (II)

Serial pseudo code for this procedure:

```
npoints = 10000
circle_count = 0

do j = 1,npoints
  generate 2 random numbers between 0 and 1
  xcoordinate = random1
  ycoordinate = random2
  if (xcoordinate, ycoordinate) inside circle
  then circle_count = circle_count + 1
end do

PI = 4.0*circle_count/npoints
```

→ "embarrassingly parallel" solution:

- Break the loop iterations into chunks that can be executed by different tasks simultaneously.
- Each task executes its portion of the loop a number of times.
- Each task can do its work without requiring any information from the other tasks (there are no data dependencies).
- Master task receives results from other tasks using send/receive point-to-point operations.

```
npoints = 10000
circle_count = 0

p = number of tasks
num = npoints/p

find out if I am MASTER or WORKER

do j = 1,num
  generate 2 random numbers between 0 and 1
  xcoordinate = random1
  ycoordinate = random2
  if (xcoordinate, ycoordinate) inside circle
  then circle_count = circle_count + 1
end do

if I am MASTER
  receive from WORKERS their circle_counts
  compute PI (use MASTER and WORKER calculations)
else if I am WORKER
  send to MASTER circle_count
endif
```

→ **red: highlights changes for parallelism.**

→ **Try to implement this example (probably using a reduction clause).**

5. Discrete State DP

- **Check the speed-up** for combinations of MPI processes and a variety of **the discretization level** on one node of **GRACE** (use slurm).
- Generate Speed up graphs (normalize to the 1 MPI process (=1 CPU) result).
- Ensure that the serial and parallel return the same results.